# Deep Learning Project 1

Pranav A (20478966)
Big Data Institute
Hong Kong University of Science and Technology

November 16, 2017

**Abstract**

In this project, I have used logistic regression and multilayer perceptron as the classification of the data points. At the end, two-layered multilayer perceptron outperformed than linear regression, achieving an accuracy of 97.3% on the validation set. However, the multilayer perceptron was overfitting before converging completely due to its complex architecture.

## 1 Introduction

Logistic regression is the most straightforward way to do classification. This is visualized as the weighted-sum of the features to produce the best output. This can also be thought of as a multilayer perceptron with no hidden layers.

MLP (Multilayer perceptron) is the most fundamental forms of the artificial neural networks. MLP introduces non-linearity through activation functions and extra layers to produce complex decision boundaries.

In this project, I have used these two techniques on the provided dataset.

## 2 Experimental Setup

The dataset comprises of 57 features with around 3220 items. Since MLP are prone to overfitting, I have divided the dataset into traning and validation sets. The training set comprises of 3000 items and the validation sets uses 220 items.

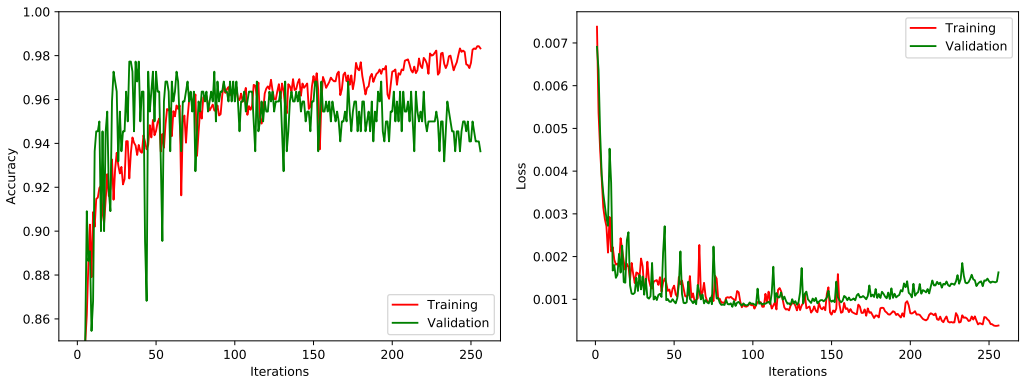


Figure 1: Plots of the learning curve for the 2-layers MLP. On the left, accuracy is plotted while on the right, loss is plotted

I have used Pytorch [Paszke et al., ] for developing this MLP architecture.

For logistic regression, I have used no hidden layers. Thus the problem of MLP reduces down to just logistic regression. Then I used Stochastic Gradient Descent, with a batch size of 128 and learning rate of 0.001. Finally I used Cross-Entropy loss function with softmax as the output.

To generate a learning curve, at every iteration I plotted the loss and accuracy for training and validation set.

For multilayer perceptron, I have used two hidden layers. Both hidden layers contain 64 neurons. I have used Adam [Kingma and Ba, 2014] as the optimizer, with a batch size of 128 and learning rate of 0.001. The activation functions used were Rectified Linear Units (ReLUs) [Dahl et al., 2013] at every layer (expect the output one). Finally I used Cross-Entropy loss function after taking softmax of the output.

# 3 Final Remarks

| | Training Accuracy | Validation Accuracy |
|---|---|---|
| Logistic Regression | 89.6% | 87.2% |
| MLP with 2 layers | **98.4%** | **97.3%** |

Table 1: Accuracy results of the algorithms

The results are shown in Table 1. Clearly, MLP with 2 layers performs better than Logistic Regression. I have also plotted the learning curves of the MLP experiment in the figure 1.

From the figures it is easy to see that model has not fully converged but starts to overfit after 150 iterations. This is due to no regularization has been applied to the MLP architecture. Maybe, a less complex MLP with less layers and dropout would have resulted in a better performance without being prone to overfitting.

# References


[Dahl et al., 2013] Dahl, G. E., Sainath, T. N., and Hinton, G. E. (2013). Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE.

[Kingma and Ba, 2014] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Paszke et al., ] Paszke, A., Chintala, S., Collobert, R., Kavukcuoglu, K., Farabet, C., Bengio, S., Melvin, I., Weston, J., and Mariethoz, J. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration, may 2017.